The time-dependent capacitated profitable tour problem with time windows and precedence constraints

Sun, Peng; Veelenturf, Lucas P.; Dabia, Said; Van Woensel, Tom

published in
European Journal of Operational Research
2018

DOI (link to publisher)
10.1016/j.ejor.2017.07.004

document version
Publisher's PDF, also known as Version of record

citation for published version (APA)
Sun, P., Veelenturf, L. P., Dabia, S., & Van Woensel, T. (2018). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research*, *264*(3), 1058-1073. https://doi.org/10.1016/j.ejor.2017.07.004

European Journal of Operational Research 264 (2018) 1058–1073

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

The time-dependent capacitated profitable tour problem with time windows and precedence constraints

Peng Sun[a,*], Lucas P. Veelenturf[a], Said Dabia[b], Tom Van Woensel[a]

[a] School of Industrial Engineering, Operations, Planning, Accounting and Control (OPAC), Eindhoven University of Technology, Eindhoven 5600 MB, The Netherlands
[b] Department of Information, Logistics and Innovation, Vrije Universiteit Amsterdam, Amsterdam 1081 HV, The Netherlands

ARTICLE INFO

Article history:
Received 16 August 2016
Accepted 3 July 2017
Available online 12 July 2017

Keywords:
Transportation
Profitable tour problem
Pickup and delivery problem
Tailored labeling algorithm
Time-dependent travel times

ABSTRACT

We introduce the time-dependent capacitated profitable tour problem with time windows and precedence constraints. This problem concerns determining a tour and its departure time at the depot that maximizes the collected profit minus the total travel cost (measured by total travel time). To deal with road congestion, travel times are considered to be time-dependent. We develop a tailored labeling algorithm to find the optimal tour. Furthermore, we introduce dominance criteria to discard unpromising labels. Our computational results demonstrate that the algorithm is capable of solving instances with up to 150 locations (75 pickup and delivery requests) to optimality. Additionally, we present a restricted dynamic programing heuristic to improve the computation time. This heuristic does not guarantee optimality, but is able to find the optimal solution for 32 instances out of the 34 instances.

1. Introduction

The effective usage of empty vehicles' space is an important opportunity to increase the efficiency of urban transportation systems and to reduce traffic congestion, fuel consumption, and pollution. Companies (e.g., Uber) can generate extra income by renting out vehicles's empty space rising in their transportation processes. Several mobile applications are developed to improve the last-mile deliveries by involving the city's residents. For instance, Roadie created an on-the-way delivery network which is an online market where people post their required shipments and where anyone can offer to execute the shipment. DHL launched a platform called MyWays, enabling individuals to deliver packages with products ordered online directly to other end consumers. For those individuals with limited transportation resources, it is important to know which parcels are profitable to collect and deliver. On the one hand, serving a request may be attractive because it generates revenue. On the other hand, there is additional cost for serving the request. Consequently, it might not always be economically beneficial to serve a request. Moreover, customers are not always available to receive the ordered goods (i.e., shops are not open 24/7 or e-commerce customers are not always home). Therefore, these customers propose specific time windows in which they want to be served. These extra constraints make it a challenge for the independent courier to be everywhere on time. He has to select a subset of the requests to be served within the requested time windows. Furthermore, due to the limited capacity of the road network and increased traffic congestion, the travel speed is serious affected by these traffic fluctuations. This results in great variations in travel times and thereby on the arrival times at customers. Therefore, the travel time between locations is dependent on the time the driver departs (e.g., in rush hours traveling takes more time). Not considering the time-dependent travel time results in too late arrivals at the customers (see e.g., Taş, Dellaert, van Woensel, & de Kok, 2014). All these complex situations illustrate the need for the independent couriers to have a tool to help them out in making decisions on which request to serve and which time to depart. However, scarce literature can be found that focus on this area.

This paper aims at building such a tool by modeling and solving a time-dependent capacitated profitable tour problem with time windows and precedence constraints. We consider a single vehicle with capacity limit and a set of requests which have a pick-up and a delivery node. Each pickup node and delivery node has its own time window in which it should be served. Moreover, a delivery node of a request can only be served after its pickup node is visited. For each served request a profit is collected. To capture travel speed variation during a day, a time-dependent travel time function is assigned to each edge linking two nodes. The objective is to determine the vehicle's tour starting and ending at the depot, and

* Corresponding author.
E-mail addresses: P.Sun@tue.nl (P. Sun), L.p.veelenturf@tue.nl (L.P. Veelenturf), S.dabia@vu.nl (S. Dabia), T.v.Woensel@tue.nl (T. Van Woensel).

http://dx.doi.org/10.1016/j.ejor.2017.07.004
0377-2217/

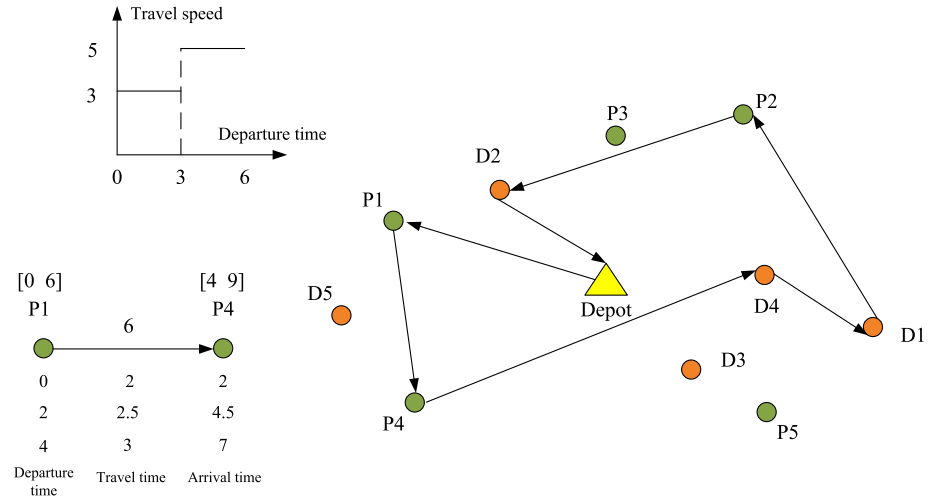

Fig. 1. An illustration of the time-dependent capacitated profitable tour problem with time windows and precedence constraints. P_i and D_i are the pickup node and delivery node of request i, respectively.

maximizing the difference between the total collected profits and total travel cost (see Fig. 1). Following the literature in this area, the tour's total travel cost is equal to the tour's duration.

The described problem is NP-hard because it is an extension of the traveling salesman problem with pickup and delivery (TSPPD), which itself is an extension of the traveling salesman problem (TSP). In contrast to the TSPPD and the TSP, in our problem, it is not necessary to visit all the requests.

The main contributions of this paper are summarized as follows. First, we introduce a new model which extends the classical TSPPD by having time-dependent travel times and the option to reject requests. Secondly, we propose an exact solution method for this problem by developing a tailored labeling algorithm in which novel and strong dominance criteria are used. Finally, a restricted dynamic programing heuristic is proposed with a high solution quality and lower computation times than the labeling algorithm.

The remainder of the paper is structured as follows. Section 2 provides a brief review of related existing work. Section 3 defines the problem and introduces a mathematical formulation of the problem. In Section 4, we present the tailored labeling algorithm and in Section 5 the restricted dynamic programing heuristic is introduced. Finally, computational results are reported in Section 6, followed by conclusions in Section 7.

2. Literature review

There are three classes of problems closely related to the problem studied in this paper: the traveling salesman problem with pickup and delivery (TSPPD), the time-dependent vehicle routing problem (TDVRP) and the traveling salesman problem with profits (TSP with profits).

2.1. The traveling salesman problem with pickup and delivery (TSPPD)

Our problem extends the TSPPD by considering time-dependent travel times. The TSPPD is firstly introduced by Ruland and Rodin (1997) and is proven to be of great use in applications like dial-a-ride systems and courier services. Although the pickup and delivery problem (PDP), in which the TSPPD can appear as a subproblem, is extensively studied in the literature, only limited research focuses on the TSPPD.

Currently, the most popular methodology for solving the TSPPD is branch-and-cut. Ruland (1994) and Ruland and Rodin (1997) considered the undirected case of this problem and developed four classes of valid inequalities that are embedded in a branch-and-cut algorithm. The algorithm is tested on instances with up to 15 pickup and delivery requests. Recently, Dumitrescu, Ropke, and Cordeau (2010) studied the same problem, the authors analyzed its polyhedral structure and proposed new valid inequalities that are shown to be facets for the TSPPD polytopes. Their algorithm is capable of solving instances with up to 35 pickup and delivery requests to optimality.

The TSPPD appears as pricing problem for the PDP and is in that situation usually named as the elementary shortest path problem with time windows, capacity and pickup and delivery (ESPPTWCPD). Sol (1994), Sigurd and Pisinger (2004) and Ropke, Cordeau, and Laporte (2009) presented labeling algorithms with several different dominance rules to solve this problem to optimality.

2.2. The time-dependent vehicle routing problem with time windows (TDVRP)

Another related problem is the TDVRP. Although the TDVRP has attracted the attention of many researchers, literature on this subject remains scarce. The pioneering work is done by Malandraki and Daskin (1992) and Malandraki and Dial (1996). In these papers mixed integer linear programs and several heuristics to solve the problem are proposed. The First-In-First-Out (FIFO) property, which implies that for every arc a later departure time results in a later (or equal) arrival time, is an intuitive and desirable property for time dependent routing problems. Ichoua, Gendreau, and Potvin (2003) and Dabia, Ropke, van Woensel, and de Kok (2013) considered the TDVRP with travel time variability modeled by "constant speed" time periods, which ensures the FIFO property. The idea of constant speed time periods is adopted in our problem as well.

Due to the complexity of the time-dependent problem, most of the existing algorithms are based on heuristics. In van Woensel, Kerbache, Peremans, and Vandaele (2008) a tabu search heuristic is used to solve the capacitated vehicle routing problem with time dependent travel times. An approximation based on queueing theory and the number of vehicles on a link is used to determine travel speeds. Donati, Montemanni, Casagrande, Rizzolo, and Gambardella (2008) developed a multi-ant colony system for the TDVRP and Ibaraki et al. (2008) proposed an iterated local search heuristic for the time-dependent vehicle routing problem with time windows (TDVRPTW). Recently, Dabia et al. (2013)

Table 1
Parameters.

Notation	Definition
$R = \{1, \ldots, n\}$	Set of requests
N	Set of nodes
A	Set of arcs
N_P	Set of pickup nodes
N_D	Set of delivery nodes
$(i, n+i)$	A transportation request R_i
r_i	Profit of request R_i
q_i	Demand of request R_i
Q	Carrying capacity of the vehicle
$[e_i, l_i]$	Time window of node i
s_i	Service time at node i
t_i	Departure time from node i
$\tau_{ij}(t_i)$	Travel time from node i to node j with departure time t_i at node i

developed a branch-and-price algorithm for the TDVRPTW, where a tailored labeling algorithm is presented to solve the time-dependent shortest path problem with resource constraint (TDSP-PRC), which is the pricing problem in the algorithm.

2.3. The traveling salesman problem with profits (TSP with profits)

The proposed problem extends the traveling salesman problem with profits (TSP with profits), in which profits are associated with each request and the overall goal is to find the shortest tour with the maximum collected profits. This means that in contrast to the original TSP, not all nodes have to be visited. In comparison with our study, it does not include time dependency and requests with pickup and delivery nodes.

According to Feillet, Dejax, and Gendreau (2005), TSPs with profits can be categorized into three generic problems depending on the way the terms *profits* and *travel time* are addressed in the objective function and constraints. They can be classified in the following categories: the profitable tour problem (PTP), the prize-collecting traveling salesman problem (PCTSP), and the orienteering problem (OP). Dell'Amico, Maffioli, and Varbrand (1995) studied the profitable tour problem (PTP) where both the profits and the travel time are combined in the objective function. Our study builds upon the PTP by having the profits and the travel time in the objective as well. In the prize-collecting TSP (PCTSP) the objective is similar to PTP but a constraint is added to ensure that a minimum amount of profits is collected within the tour. In the original definition of the PCTSP by Balas (1989) there were also penalty values for unvisited nodes within the objective function. An abundant number of publications is devoted to the orienteering problem (OP), which aims to maximize the collected profits subject to a constraint on the maximum allowed tour length. This problem is also known as the selective traveling salesman problem (Laporte & Martello, 1990).

A variant of the OP is the time-dependent orienteering problem (TDOP) which includes time-dependent travel times. Fomin and Lingas (2002) provided a $(2+\epsilon)$-approximation algorithm for the TDOP which runs in polynomial time if the ratio between the minimum and maximum travel time between any two sites is constant. Li (2011) designed a novel dynamic labeling algorithm for the TDOP in which time is measured in discrete units. Therefore, the FIFO property may not be satisfied in their model. Verbeeck, Aghezzaf, and Vansteenwegen (2014) provided a fast solution method for the TDOP based on an ant colony optimization algorithm. Recently, Verbeeck, Vansteenwegen, and Aghezzaf (2016) presented an ant colony optimization based algorithm for the stochastic variant of the TDOP, which is addressed as the stochastic time-dependent orienteering problem with time windows. For more details about the OP and its variants, readers are referred to Vansteenwegen, Souffriau, and Oudheusden (2011) and Gunawan, Lau, and Vansteenwegen (2016).

To the best of our knowledge, no literature is found that handles a combination of precedence in pickup and delivery, profit-maximizing selection and time-dependent travel time routing cost minimization at the same time. Thus, in this study, we introduce the time-dependent capacitated profitable tour problem with time windows and precedence constraints, which takes care of these three challenges simultaneously. Moreover, both exact and heuristic methods are proposed to solve this problem.

3. Problem description and mathematical formulation

In this section, we first define the problem and introduce the notation used throughout the paper. Afterwards, we present a mathematical formulation for the problem.

3.1. Problem definition

The time-dependent capacitated profitable tour problem with time windows and precedence constraints is defined as follows. We consider a set of n requests $R_1, \ldots, R_n$, where R_i $(i = 1, \ldots, n)$ is associated with the pickup node i and the corresponding delivery node $n+i$. Let $G = (N, A)$ be a directed graph, where $N = \{0, 1, \ldots, 2n+1\}$ is the set of all nodes, and 0 and $2n+1$ represent the origin and destination depot of the vehicle. We define the subsets $N_P = \{1, \ldots, n\}$ and $N_D = \{n+1, \ldots, 2n\}$ as the pickup and delivery nodes, respectively. With each pickup node $i \in N_P$ a profit r_i and a load q_i are associated, and with each delivery node a load q_{n+i} is associated. For the requests, it must hold that $q_i = -q_{n+i}$. There is no inventory at the depots and therefore $q_0 = q_{2n+1} = 0$. To serve the requests we have one vehicle available with limited capacity Q.

A hard time window $[e_j, l_j]$ is associated with each node $j \in N_P \cup N_D$, where e_j and l_j represent the earliest and latest time, respectively, at which the service at node j may start. The service time is denoted by s_j. A vehicle needs to wait until time e_j, if it is arriving at node j before time e_j; and arriving later than l_j is not allowed. We denote $[e_0, l_0]$, $[e_{2n+1}, l_{2n+1}]$ as the time windows of the origin and the destination depot, respectively. Without loss of generality, we assume that $e_0 = 0$ and $s_0 = s_{2n+1} = 0$.

Let $\tau_{ij}(t_i)$ denote the travel time from node i to node j, which depends on the departure time t_i at node i. Then, we can define the set of feasible arcs as $A = \{(i, j) \in N \times N : i \neq j \text{ and } e_i + s_i + \tau_{ij}(e_i + s_i) \leq l_j\}$. This means that an arc from node i to node j is only included if it is possible to go from node i to j while respecting the time windows of both nodes.

The notation is summarized in Table 1.

The planning horizon is divided into several time periods. Each arc $(i, j) \in A$ has a speed profile associated with it, which consists of

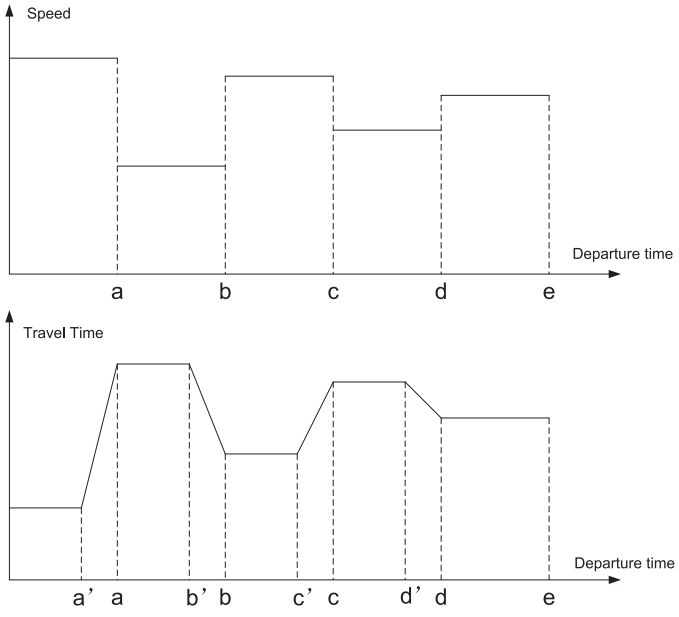

Fig. 2. Speed and travel time functions.

a constant speed within each time period. By using those stepwise speed functions, the FIFO property holds for every arc in the graph G (i.e., a later departure always leads to a later arrival and therefore overtaking will not occur). The speed profiles can be different for each arc.

Fig. 2 depicts a speed profile and the corresponding travel time function for some arc (i, j). Using the idea described in Ichoua et al. (2003), we denote the points a, b, c, d, and e where the speed changes as *speed breakpoints*. There are also *travel time breakpoints* in the travel time function. These are the departure times which ensure an arrival at node j exactly at the time of a speed breakpoint (e.g., a' is the departure time at node i to arrive at node j at time a).

The travel time function is piecewise linear and can be represented by the breakpoints values. Note that in case of time-dependent travel times, the triangle inequality does not necessarily hold. Intuitively, when the direct link between node h and l is heavily congested, we may reach destination node l earlier by taking a diverted routed (i.e., via one or several other nodes) than by the direct link from node h.

Because of the FIFO property of the travel time functions, a later departure at the depot 0 always results in a later arrival time at node i. Therefore, if a path is infeasible for a certain departure time t_0 at the origin depot (i.e., a time window of a node in the path is violated), it will also be infeasible for any departure time $t' \geq t_0$ at the origin depot. Given a path $p = (v_0, v_1, \ldots, v_k)$ with $v_0 = 0$ and v_i being the node at position i in the path p, we define $\delta^p_{v_i}(t)$ as the ready time function at node v_i in path p given a departure time t at node 0. This ready time function is nondecreasing in t and can be calculated recursively for each node in the path as follows:

$$\delta^p_{v_i}(t) = \begin{cases} t & \text{if } i = 0, \\ \max\{e_{v_i} + s_{v_i}, \delta^p_{v_{i-1}}(t) \\ \quad +\tau_{v_{i-1},v_i}(\delta^p_{v_{i-1}}(t)) + s_{v_i}\} & \text{otherwise.} \end{cases} \tag{1}$$

The ready time function is piecewise linear and this means that we can represent the ready time function by using the *ready time function breakpoints*. These are the breakpoints of the ready time function of the predecessor node, breakpoints of the travel time function, and the boundary values of the time window of node v_i.

The duration of the path given a departure time t at node 0 can be calculated as $\delta^p_{v_k}(t) - t$, which is again a piecewise linear function. In this problem we minimize the total duration of the selected tour instead of the sum of the arc cost. As the duration is a piecewise linear function of the departure time, it is clear that the minimum duration of a tour can be computed by only considering the breakpoints of the ready time function.

3.2. Mathematical formulation

For every arc $(i, j) \in A$, we denote T_{ij} as the set of time periods of the corresponding travel time function $\tau_{ij}(t_i)$. A time period $T_m \in T_{ij}$, is defined by two consecutive travel time breakpoints, $T_m = [w_m, w_{m+1}]$. As $\tau_{ij}(t_i)$ is linear in each time period, using w_m, w_{m+1}, $\tau_{ij}(w_m)$ and $\tau_{ij}(w_{m+1})$, we can easily calculate the corresponding slope θ_m and its intersection η_m with the y-axis. Therefore

$$\tau_{ij}(t_i) = \theta_m t_i + \eta_m. \qquad \forall t_i \in T_m \tag{2}$$

Furthermore, let x^m_{ij} be a binary variable that takes value 1 if and only if the vehicle traverses the arc $(i, j) \in A$ with a departure time in time period m. A variable t^m_{ij} is introduced to denote this departure time of traveling from i to j in time period T_m. This means that t^m_{ij} is such that

$$t^m_{ij} = \begin{cases} t_i & \text{if } x^m_{ij} = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Consequently, when traveling from i to j, we have that:

$$t_i = \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{ij}|} t^m_{ij}. \tag{4}$$

It means that the travel time function $\tau_{ij}(t_i)$ of arc (i, j) at node i can be written as:

$$\tau_{ij}(t_i) = \sum_{m=0}^{|T_{ij}|} (\theta_m t^m_{ij} + \eta_m x^m_{ij}). \tag{5}$$

Let y_i be a binary variable that equals 1 if and only if node $i \in N_P \cup N_D$ is visited. Furthermore, let Q_i, $i \in N$ be a nonnegative integer variable that is the load of the vehicle upon departure from node i. Then the mixed integer programing formulation is given as follows:

$$\max \sum_{j \in N_P} r_j y_j - (t_{2n+1} - t_0) \tag{6}$$

subject to

$$\sum_{j \in N_P} \sum_{m=0}^{|T_{0j}|} x^m_{0j} = 1 \tag{7}$$

$$\sum_{i \in N_D} \sum_{m=0}^{|T_{i,2n+1}|} x^m_{i,2n+1} = 1 \tag{8}$$

$$\sum_{i \in N \setminus \{2n+1\}} \sum_{m=0}^{|T_{ij}|} x^m_{ij} = y_j \qquad \forall j \in N \setminus \{0\} \tag{9}$$

$$\sum_{i \in N \setminus \{2n+1\}} \sum_{m=0}^{|T_{ik}|} x^m_{ik} - \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{kj}|} x^m_{kj} = 0 \qquad \forall k \in N \setminus \{0, 2n+1\} \tag{10}$$

$$\sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{ij}|} x^m_{ij} - \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{n+i,j}|} x^m_{n+i,j} = 0 \qquad \forall i \in N_P \tag{11}$$

$$t_j \geq (1+\theta_m)t_{ij}^m + \eta_m x_{ij}^m + s_j x_{ij}^m \qquad \forall i \in N \setminus \{2n+1\}, j \in N \setminus \{0\} \tag{12}$$

$$Q_i + q_j \leq Q_j + M(1 - x_{ij}^m) \qquad \forall i, j \in N, \forall m, m+1 \in |\, T_{ij} \,| \tag{13}$$

$$t_{n+i} \geq t_i \qquad \forall i \in N_P \tag{14}$$

$$t_i = \sum_{j \in N \setminus \{0\}} \sum_{m=0}^{|T_{ij}|} t_{ij}^m \qquad \forall i \in N \setminus \{2n+1\} \tag{15}$$

$$w_m x_{ij}^m \leq t_{ij}^m \leq w_{m+1} x_{ij}^m \qquad \forall i, j \in N, \forall m, m+1 \in |\, T_{ij} \,| \tag{16}$$

$$e_i y_i \leq t_i \leq l_i y_i \qquad \forall i \in N_P \cup N_D \tag{17}$$

$$\max\{0, q_i\} \leq Q_i \leq \min\{Q, Q + q_i\} \qquad \forall i \in N \tag{18}$$

$$x_{ij}^m, y_i \in \{0, 1\} \qquad \forall i, j \in N, \forall m \in |\, T_{ij} \,| \tag{19}$$

The objective function (6) aims to find a tour that maximizes the collected profits minus the total traveling duration. Constraints (7)–(8) guarantee that the path starts at the origin depot 0 and ends at the destination depot $2n+1$. Constraints (9) guarantee that every node, except the nodes representing the start and end depots, is visited at most once. Constraints (10) keep the flow conservation. Constraints (11) ensure that it is not possible to visit only the pickup node or only the delivery node of a certain request. Constraints (12) guarantee that the departure time at a node in the route is larger or equal than the sum of the departure time from the previous node, the travel time between these two nodes and the required service time. Constraints (13) determine the consistency of the load variables. Precedence constraints (14) ensure that for each request i, the pickup node is visited before the delivery node. Constraints (15) are formulated as mentioned before in (4). Constraints (16) and (17) force the departure time of each request to be in the given time period and the given time window. Finally, Constraints (18) ensure that the load in the vehicle is never larger than the capacity of the vehicle.

Due to the computational inefficiency of solving large-scale instances with a commercial ILP solver, we develop a tailored labeling algorithm to solve this problem.

4. A tailored labeling algorithm

In order to solve our problem, we introduce a new exact dynamic programing algorithm, that is named as the tailored labeling algorithm. Ropke et al. (2009) developed a labeling algorithm to solve the pickup and delivery problems with time windows (PDPTW). However, this time-independent algorithm is only efficient if the triangle inequality holds. More recently, Dabia et al. (2013) proposed another labeling algorithm for the time-dependent vehicle routing problem with time windows. Their algorithm has great potential for the time-dependent routing problem without precedence constraints.

In our situation the triangle inequality does not necessarily hold due to the time dependent travel times and furthermore precedence constraints are present. Therefore, we need to develop a new algorithm. Note that the proposed algorithm can be generalized to solve other time-dependent routing problems with precedence constraints.

The algorithm starts generating labels from the depot 0. It progressively extends all feasible labels until they reach the end depot $2n+1$. Moreover, to speed up our tailored labeling algorithm, instead of starting the label extension only from the origin depot 0 in a forward direction, we simultaneously generate labels in backward direction from the destination depot to its predecessors as well. Where both forward labels and backward labels are extended to some time t_m (e.g., the middle of the planning horizon) but not further. At the end, complete paths are generated by merging the partial paths of forward and backward labels. All complete paths are evaluated and the path with the best objective function value is the optimal path. This bidirectional approach has shown great potential for improving the running time of related resource constrained shortest path problems (see, e.g., Righini and Salani, 2006; Dabia et al., 2013).

The forward labeling algorithm is introduced in Section 4.1, followed by the backward labeling algorithm in Section 4.2. If labels are dominated by the criterion introduced in Sections 4.1.3 and 4.2.3, they are removed from the list. Note that if in the procedure none of the labels are dominated, this algorithm is equal to the complete enumeration of all feasible paths. Therefore the dominance criterion is very important. Finally, we discuss the way to merge the partial paths of forward and backward labels in Section 4.3.

4.1. The forward labeling algorithm

In the forward labeling algorithm we start generating labels from the start depot. The definition of a forward label is discussed in Section 4.1.1. The labels are extended if the extension is feasible as discussed in Section 4.1.2. The dominance criterion for forward labels is discussed in Section 4.1.3.

4.1.1. Forward label

For each forward label L_f, we use the following notation:

$p(L_f)$	The partial path of label L_f.
$v(L_f)$ *	The last node visited on the partial path $p(L_f)$.
$L^{-1}(L_f)$ *	The parent label from which L_f originates by extending it with $v(L_f)$.
$O(L_f)$ *	The set of incomplete requests in $p(L_f)$, i.e., the pickup node is visited but not the delivery node.
$U(L_f)$ *	The set of requests for which the pickup nodes are already visited along the partial path $p(L_f)$. It contains both the complete and the incomplete requests. Therefore, $O(L_f) \subseteq U(L_f)$.
$P(L_f)$	The set of pickup nodes not visited in $p(L_f)$, i.e., $j \in N_P$ and $R_j \notin U(L_f)$.
$D(L_f)$	The set of delivery nodes of incomplete requests in $p(L_f)$, i.e., $j \in N_D$ and $R_{j-n} \in O(L_f)$.
$q(L_f)$ *	The load of the vehicle after visiting node $v(L_f)$.
$\delta_{L_f}(t)$ *	The piecewise linear function that represents the ready time at $v(L_f)$ if the vehicle departed at the origin depot at t and reached $v(L_f)$ through partial path $p(L_f)$. Moreover, $\delta_{L_f}(0)$ is the earliest ready time at $v(L_f)$ since the earliest departure time at the origin depot is 0.
$r(L_f)$ *	The overall profits collected by serving the requests visited on the partial path $p(L_f)$.

Only the items marked with a * are stored in the label. The set $D(L_f)$ and $P(L_f)$ can be deduced from the sets $O(L_f)$ and $U(L_f)$. Furthermore, the partial path can be deduced from iteratively checking the last node visited in the parent label of which the this label was an extension.

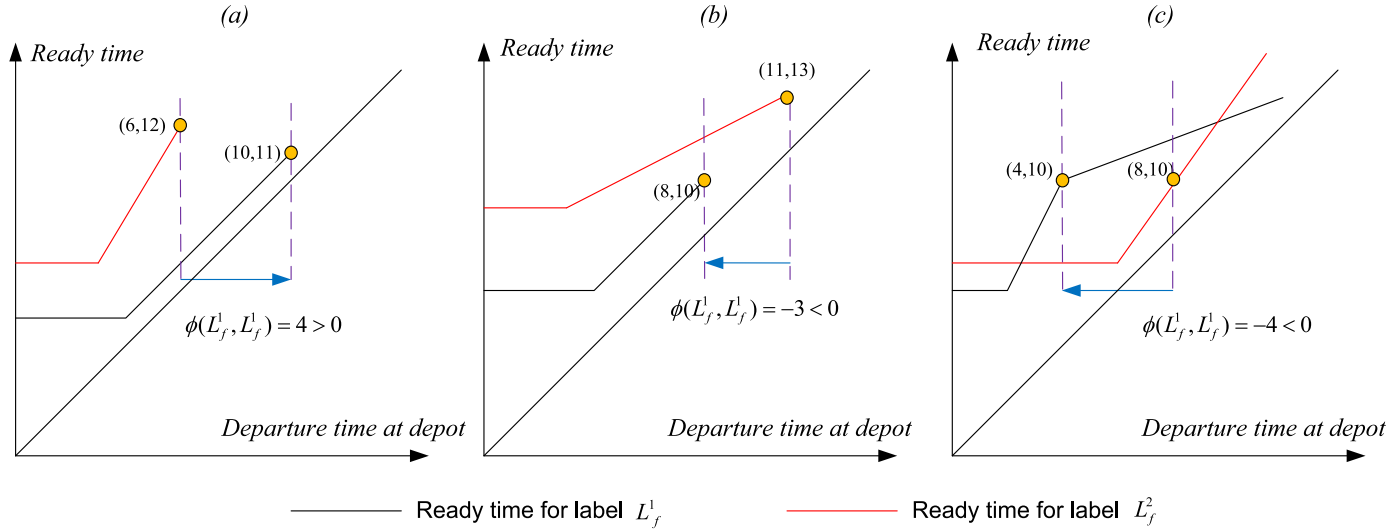

Fig. 3. Illustration of $\phi(L_f^1, L_f^2)$.

4.1.2. Label extension

We extend a label L_f' along an arc $(v(L_f'), j)$, only when the extension is feasible in terms of time windows and capacity. First, the following two conditions should be met:

$$\delta_{L_f'}(0) + \tau_{v(L_f'),j}(\delta_{L_f'}(0)) + s_j \le \min\{t_m, l_j + s_j\} \quad \wedge \quad j \in N \setminus \{0\} \tag{20}$$

$$q(L_f) + q_j \le Q \quad \wedge \quad j \in N \setminus \{0\} \tag{21}$$

Condition (20) ensures that an extension to node j can only be performed if node j can be reached within its time window and guarantees that the extension is stopped before t_m is exceeded. Condition (21) ensures that an extension to node j is only possible if there is enough capacity to deal with the load of node j.

Secondly, L_f' and j must also satisfy one of the following three conditions:

$$j \notin U(L_f') \quad \wedge \quad j \in N_P \tag{22}$$

$$j - n \in O(L_f') \quad \wedge \quad j \in N_D \tag{23}$$

$$O(L_f') = \emptyset \quad \wedge \quad j = 2n + 1 \tag{24}$$

Condition (22) states that j should not have been visited before, if it is a pickup node. Condition (23) indicates that if j is a delivery node, the corresponding pickup node should have been visited already. The last condition, Condition (24), states that if j is the end depot then all visited requests should have been completed. In the presence of those conditions, only elementary paths that satisfy precedence constraint (11) are generated.

At last, we need to check that all delivery nodes of requests for which the pickup node is already visited in $p(L_f')$ can still be reached. In case the triangle inequality holds, a node is unreachable if traversing the direct arc from j to this node is not possible by capacity, time window or precedence constraints. However, time-dependent travel times cannot guarantee the triangle inequality. Therefore, a node that is unreachable via the direct arc from node j by the time window constraints might still be reachable indirectly via a diverted route. First, we need to know the earliest ready time at node j after following partial path $p(L_f')$ before visiting node j, which will be denoted by $t_r(L_f', j)$. It holds that $t_r(L_f', j) = \max\{e_j + s_j, \delta_{L_f'}(0) + \tau_{L_f',j}(\delta_{L_f'}(0)) + s_j\}$. Then, we need for any unvisited node k the earliest arrival time given that the vehicle visits node j and k consecutively after partial path $p(L_f')$, which can be computed by $t_r(L_f', j) + \tau_{jk}(t_r(L_f', j))$. Finally, the earliest possible time the vehicle could reach a node after node j is given by $t_e(L_f', j) = \min_{k \in P(L_f') \cup D(L_f')}\{t_r(L_f', j) + \tau_{jk}(t_r(L_f', j))\}$. This means that any node k with the latest allowed arrival time l_k earlier than this time (i.e., $l_k < t_e(L_f', j)$) is unreachable from j, also in an indirect way as no node could be reached before $t_e(L_f', j)$. If a delivery node k is unreachable after j and its corresponding pickup node is already visited (i.e., $k - n \in O(L_f')$), the extension to j is not feasible as the picked up item cannot be delivered anymore. Therefore, as stated in condition (25), all delivery nodes of requests of which the pickup node is already visited in $p(L_f')$ should still be reachable to make sure that extending label L_f' to j is feasible. Note that this test can be done quickly, but we might fail to find all unreachable delivery nodes.

$$l_k \ge t_e(L_f', j) \quad \forall k \in D(L_f') : k \ne j \tag{25}$$

If the extension along the arc $(v(L_f'), j)$ is feasible according to all described conditions, then a new label L_f is created. The information in label L_f is updated as follows:

$$L^{-1}(L_f) = L_f' \tag{26}$$

$$v(L_f) = j \tag{27}$$

$$\delta_{L_f}(t) = \max\{e_j + s_j, \delta_{L_f'}(t) + \tau_{L_f',j}(\delta_{L_f'}(t)) + s_j\} \tag{28}$$

$$q(L_f) = q(L_f') + q_j \tag{29}$$

$$r(L_f) = \begin{cases} r(L_f') + r_j & \text{if } j \in N_P, \\ r(L_f') & \text{otherwise.} \end{cases} \tag{30}$$

$$O(L_f) = \begin{cases} O(L_f') \cup \{j\} & \text{if } j \in N_P, \\ O(L_f') \setminus \{j - n\} & \text{if } j \in N_D. \end{cases} \tag{31}$$

$$U(L_f) = \begin{cases} U(L_f^{'}) \cup \{j\} & \text{if } j \in N_P, \\ U(L_f^{'}) & \text{otherwise.} \end{cases} \quad (32)$$

Eqs. (26) –(30) set the last visited node, the ready time function, the load, and the collected profits of the new label, respectively. Eq. (31) updates the set of incomplete requests $O(L_f)$ and Eq. (32) updates the set of visited pickup nodes $U(L_f)$.

4.1.3. Label dominance

Let $dom(L_f)$ and $img(L_f)$ be the domain and image of the ready time function $\delta_{L_f}(t)$, respectively. If the partial path is feasible, a departure at time 0 from the origin depot is always feasible. Therefore, $dom(L_f)$ is always of the form $[0, t]$ for some $t \geq 0$. When $v(L_f) = 2n+1$, the objective function value of the path corresponding to L_f is:

$$obj(L_f) = r(L_f) - \min_{t \in dom(L_f)} \{\delta_{L_f}(t) - t\} \quad (33)$$

In the labeling algorithm, all possible extensions are processed and stored for each label. However, the number of labels that can be processed is typically very large and computationally expensive. Therefore, a dominance test is established between pairs of labels that have the same last visited node. The number of labels is reduced by only storing the non-dominated labels. Before the dominance criterion is introduced some definitions need to be provided.

First, similar to the idea of Feillet, Dejax, and Gendreau (2004), we introduce for every label L_f the set $\widetilde{U}(L_f)$ which extends $U(L_f)$ by adding requests of which the pickup node is unreachable from $v(L_f)$. Similar to the discussion in Section 4.1.2 it can be derived that the earliest possible time the vehicle could reach a pickup node after $v(L_f)$ is given by $t_e(L_f) = \min_{j \in P(L_f) \cup D(L_f)} \{\delta_{L_f}(0) + \tau_{v(L_f)j}(\delta_{L_f}(0))\}$. This means that any pickup node j with the latest allowed arrival time l_j earlier than this time (i.e., $l_j < t_e(L_f)$) is unreachable from $v(L_f)$. Therefore, the corresponding request cannot be served anymore and can be added to the set $\widetilde{U}(L_f)$. Note that the check $l_j < t_e(L_f)$ does not guarantee to find all unreachable pick up nodes.

Secondly, we define the interval

$$I \subseteq (-\infty, \max(dom(L_f^1)) - \max(dom(L_f^2))). \quad (34)$$

Based on I, we also define a real number $\phi(L_f^1, L_f^2)$,

$$\phi(L_f^1, L_f^2) = \max\{x \in I : \delta_{L_f^1}(\max\{0, t+x\}) \leq \delta_{L_f^2}(t), \quad \forall t \in dom(L_f^2)\}. \quad (35)$$

When $\phi(L_f^1, L_f^2)$ is positive, it indicates that the vehicle can depart at maximum $\phi(L_f^1, L_f^2)$ time units later when traversing partial path $p(L_f^1)$ instead of traversing $p(L_f^2)$, to still reach node $v(L_f^1)$ earlier via path $p(L_f^1)$ then via partial path $p(L_f^2)$. When it is negative, it indicates that the vehicle should depart at least $\phi(L_f^1, L_f^2)$ time units earlier when traversing partial path $p(L_f^1)$ instead of traversing $p(L_f^2)$, to be able to reach node $v(L_f^1)$ earlier via path $p(L_f^1)$ then via partial path $p(L_f^2)$.

In Fig. 3, we depict several simple examples: if there is no intersection between labels L_f^1 and L_f^2, $\phi(L_f^1, L_f^2)$ is positive when $\max(dom(L_f^1)) > \max(dom(L_f^2))$ (see Fig. 3(a)), or negative when $\max(dom(L_f^1)) < \max(dom(L_f^2))$ (see Fig. 3(b)). Otherwise, $\phi(L_f^1, L_f^2)$ can only be negative (see Fig. 3(c)).

Finally, the dominance test is stated in Proposition 4.1 as follows:

Proposition 4.1. *Label L_f^2 is dominated by label L_f^1 if*

1. $v(L_f^1) = v(L_f^2)$
2. $U(L_f^1) \subseteq \widetilde{U}(L_f^2)$
3. $O(L_f^1) = O(L_f^2)$
4. $\delta_{L_f^1}(0) \leq \delta_{L_f^2}(0)$
5. $r(L_f^1) \geq r(L_f^2) - \phi(L_f^1, L_f^2)$
6. $q(L_f^1) \leq q(L_f^2)$

Proof of Proposition 4.1. Consider two labels L_f^1 and L_f^2 that satisfy the six conditions in Proposition 4.1. We need to show that (i) any feasible extension L that extends $p(L_f^2)$ to $2n+1$ is also a feasible extension for $p(L_f^1)$ to $2n+1$ and (ii) that for all these feasible extensions L it holds that $obj(L_f^1 \oplus L) \geq obj(L_f^2 \oplus L)$, where $L_f \oplus L$ is the label resulting from extending L_f with L.

With regards to point (i), first, capacity will not be violated along the path $p(L_f^1 \oplus L)$ as it was not violated on path $p(L_f^2 \oplus L)$ and by condition 6 it holds that $q(L_f^1) \leq q(L_f^2)$. Secondly, the path $p(L_f^1 \oplus L)$ is elementary. By conditions 2 and 3, all nodes visited in $p(L_f^1)$ are either nodes visited in $p(L_f^2)$ or nodes which could not be reached by any extension of label L_f^2 (i.e., the nodes of requests included in $\widetilde{U}(L_f^2) \setminus U(L_f^2)$). A feasible extension of L_f^2 cannot contain any node visited along path $p(L_f^2)$ or node which is unreachable from label L_f^2. Therefore, all nodes visited along path $p(L_f^1)$ are not visited in L, so path $p(L_f^1 \oplus L)$ is elementary as well. Third, there exists a departure time for path $p(L_f^1 \oplus L)$ which does not violate time windows. As L is a feasible extension of L_f^2 it means that there is a departure time at $v(L_f^1)$ after $\delta_{L_f^2}(0)$ making sure that all nodes in L are visited within their time windows. If condition 4 is met, the vehicle is via path $p(L_f^1)$ always able to reach $v(L_f^1)$ before this time. For example by departing at 0, the vehicle arrives at $v(L_f^1)$ at time $\delta_{L_f^1}(0)$ which is by condition 4 smaller than or equal to $\delta_{L_f^2}(0)$. Therefore, a departure at 0 over path $p(L_f^1 \oplus L)$ does not violate any time windows. In conclusion, any extension L of $p(L_f^2)$ to $2n+1$ will be a feasible extension of $p(L_f^1)$ to $2n+1$ as it results in an elementary path with does not violate time windows and capacity constraints.

Then for (ii), it still has to be proven that for all feasible extensions of L of $p(L_f^2)$ to $2n+1$ it holds that $obj(L_f^1 \oplus L) \geq obj(L_f^2 \oplus L)$. Let $L_f^{1*} = L_f^1 \oplus L$ and $L_f^{2*} = L_f^2 \oplus L$. We also denote $t_0^2 = argmin_{t \in dom(L_f^{2*})} \{\delta_{L_f^{2*}}(t) - t\}$ as the optimal departure time from the depot for path $p(L_f^{2*})$ and $r(L)$ as the sum of the profits associated with the nodes visited along path $p(L)$. The objective value of the path is:

$$\begin{aligned} obj(L_f^{2*}) &= r(L_f^{2*}) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) \\ &= r(L_f^2) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) \end{aligned} \quad (36)$$

Now consider the path $p(L_f^{1*})$ resulting from extending L_f^1 by L. Moreover, consider a departure time at the depot of this path of $t_0^1 = \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}$. The time t_0^1 is a feasible departure time for label L_f^{1*} because a departure time of 0 is always possible (as the extension of L_f^1 by L is feasible) and by the definition of $\phi(L_f^1, L_f^2)$ in Eq. (35) $t_0^2 + \phi(L_f^1, L_f^2)$ belongs to $dom(L_f^1)$ if it is non-negative. This departure time t_0^1 ensures that we reach node $v(L_f^1)$ at time $\delta_{L_f^2}(t_0^2)$ or earlier, meaning:

$$\delta_{L_f^1}(t_0^1) \leq \delta_{L_f^2}(t_0^2). \quad (37)$$

Moreover, as t_0^1 is a feasible departure time it can be used to compute a lower bound on $obj(L_f^{1*})$:

$$obj(L_f^{1*}) \geq r(L_f^{1*}) - (\delta_{L_f^{1*}}(t_0^1) - t_0^1)$$
$$= r(L_f^1) + r(L) - (\delta_{L_f^{1*}}(t_0^1) - \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}) \quad (38)$$

$$\geq r(L_f^1) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - \max\{0, t_0^2 + \phi(L_f^1, L_f^2)\}) \quad (39)$$

$$\geq r(L_f^1) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2 - \phi(L_f^1, L_f^2)) \quad (40)$$

$$\geq r(L_f^2) - \phi(L_f^1, L_f^2) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2 - \phi(L_f^1, L_f^2)) \quad (41)$$

$$\geq r(L_f^{2*}) + r(L) - (\delta_{L_f^{2*}}(t_0^2) - t_0^2) = obj(L_f^{2*}) \quad (42)$$

Note that in inequality (39) we use the property derived at (37). Inequality (40) is derived by the simple fact that $\forall x \in R : -\max\{0, x\} \leq -x$, and inequality (41) uses condition 5 of Proposition 4.1. □

4.2. The backward labeling algorithm

In the backward labeling algorithm we start in the opposite direction and start generating labels from the end depot. The definition of a backward label is discussed in Section 4.2.1. The labels are extended as discussed in Section 4.2.2. The dominance criterion for backward labels is discussed in Section 4.2.3.

4.2.1. Backward label

In the backward labeling algorithm, labels are extended from the end depot $2n+1$ to its predecessors. For a label L_b, we associate the following components:

$p(L_f)$	The partial path of label L_b.
$v(L_b)$ *	The first node visited on the partial path $p(L_b)$.
$L^{-1}(L_b)$ *	The parent label from which L_b originates by extending it with $v(L_b)$.
$O(L_b)$ *	The set of incomplete requests, i.e., the delivery is visited but not the pickup node.
$U(L_b)$ *	The set of requests for which the delivery nodes are already visited along the partial path $p(L_b)$. It contains both the complete and incomplete requests. Therefore, $O(L_b) \subseteq U(L_b)$.
$P(L_b)$	The set of pickup nodes of incomplete request in $p(L_b)$, i.e., $j \in N_P$ and $R_{j+n} \in O(L_b)$.
$D(L_b)$	The set of delivery nodes not visited in $p(L_b)$, .i.e., $j \in N_D$ and $R_{j-n} \notin U(L_b)$.
$q(L_b)$ *	The load of the tour after visiting node $v(L_b)$.
$\delta_{L_b}(t)$ *	The arrival time at the end node $2n+1$ through the partial path represented by L_b when leaving node $v(L_b)$ at time t.
$r(L_b)$ *	The overall profits collected with the requests completed on the partial path $p(L_b)$.

Again, only the items marked with a * are stored in the label and the sets $D(L_b)$ and $P(L_b)$ can be deduced from the sets $O(L_b)$ and $U(L_b)$. Furthermore, the partial path can be deduced from iteratively checking the first node visited in the parent label of which the this label was an extension.

4.2.2. Label extension

Let $dom(L_b)$ be the domain of the function $\delta_{L_b}(t)$ and let $t_l(L_b)$ denote the latest possible ready time at $v(L_b)$: $t_l(L_b) = \max(dom(L_b))$. We extend a label L_b' along an arc $(j, v(L_b'))$ to create a new label L_b. To be a feasible extensions at least the following two conditions should be met:

$$t_l(L_b) \geq \max\{t_m, e_j + s_j\} \quad \wedge \quad j \in N \setminus \{2n+1\} \quad (43)$$

$$q(L_b) + q_j \leq Q \quad \wedge \quad j \in N \setminus \{2n+1\} \quad (44)$$

Condition (43) ensures that node j can be reached within its time window and that the extension will be stopped before t_m is exceeded, while condition (44) ensures capacity feasibility. Furthermore, L_b' and j must satisfy one of the following three conditions:

$$j + n \in O(L_b') \quad \wedge \quad j \in N_P \quad (45)$$

$$j \notin U(L_b') \quad \wedge \quad j \in N_D \quad (46)$$

$$O(L_f') = \emptyset \quad \wedge \quad j = 0 \quad (47)$$

Condition (45) indicates that if j is a pickup node, the corresponding delivery node should have been visited already. Furthermore, condition (46) states that if j is a delivery node, it should not have been visited before. Finally, condition (47), states that if j is the begin depot then all visited requests should have been completed. In the presence of those conditions, only elementary paths that satisfy precedence constraint (11) are generated.

At last, it needs to be checked that all pickup nodes of incomplete requests for which the delivery node is included in $p(L_b')$ can be visited before node j. To do so, first the latest possible arrival time at node j which ensures a ready time of $t_l(L_b)$ at $v(L_b)$ should be determined. Let this be denoted by $t_r(j, L_b')$, then it holds that $t_r(j, L_b') = \min\{l_j, \max\{t : t + s_j + \tau_{jv(L_b')}(t) + s_{v(L_b')} \leq t_l(L_b')\}$. As shown $t_r(j, L_b')$ is determined by the latest possible arrival time at j (i.e., l_j) or by the latest possibility departure time to reach $v(L_b')$ on time.

Then, all nodes from which the vehicle cannot depart before time $t_r(j, L_b')$ due to time window constraints, cannot be visited before node j and are defined as unreachable. This can be made stronger by considering the travel time to node j as well. However, again due to the absence of the triangle inequality and the time dependent travel times we cannot simply consider the travel time of the direct connection. Therefore, we need for any unvisited node k the latest possible departure time to arrive at j at time $t_r(j, L_b')$, which can be computed by $\max\{t : t + \tau_{kj}(t) \leq t_r(j, L_b'))\}$. Finally, the latest possible time the vehicle could depart from any node before node j is given by $t_d(j, L_f') = \max_{k \in P(L_f') \cup D(L_f')}\{\max\{t : t + \tau_{kj}(t) \leq t_r(j, L_b')\}\}$.

This means that any node k with the earliest allowed arrival time e_k later than this time (i.e., $l_k > t_d(j, L_f')$) cannot be a predecessor of node j, also not in an indirect way as no node could be left after $t_d(j, L_f')$ and still reaching node j on time.

If a pickup node k cannot be a predecessor of node j and its corresponding delivery node is already visited (i.e., $k + n \in O(L_f')$), the extension to j is not feasible as the item which should be delivered cannot be picked up anymore. Therefore, as stated in condition (48), all pickup nodes of requests of which the delivery node is already visited in $p(L_b')$ should be a possible predecessor of j to make sure that extending label L_b' to j is feasible.

$$l_k \leq t_d(j, L_b') \quad \forall k \in P(L_b') : k \neq j \quad (48)$$

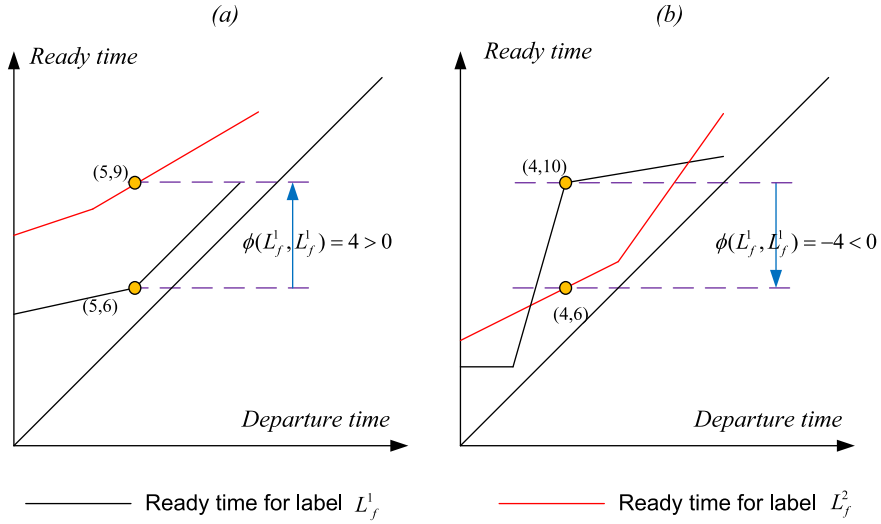

Fig. 4. Illustration of $\phi(L_f^1, L_f^2)$.

If the extension along the arc (j, L_b') is feasible according to the provided conditions, the information in label L_b is set as follows:

$$L^{-1}(L_b) = L_b' \tag{49}$$

$$v(L_b) = j \tag{50}$$

$$\delta_{L_b}(t) = \delta_{L_b'}(\max\{e_{v(L_b')}, t + \tau_{jv(L_b')}(t)\} + s_{v(L_b')}) \tag{51}$$

$$q(L_b) = q(L_b') + q_j \tag{52}$$

$$r(L_b) = \begin{cases} r(L_b') + r_j & \text{if } j \in N_P, \\ r(L_b) & \text{otherwise.} \end{cases} \tag{53}$$

$$O(L_b) = \begin{cases} O(L_b') \setminus \{j\} & \text{if } j \in N_P, \\ O(L_b') \cup \{j - n\} & \text{if } j \in N_D. \end{cases} \tag{54}$$

$$U(L_b) = \begin{cases} U(L_b') \cup \{j - n\} & \text{if } j \in N_D, \\ U(L_b) & \text{otherwise.} \end{cases} \tag{55}$$

4.2.3. Label dominance

Dominance of the backward algorithm can be constructed in the same way as in the case of the forward algorithm, because the arrival time functions are non-decreasing and stepwise linear as before.

Similar to the forward algorithm, in Proposition 4.2, we extend the set $U(L_b)$ to $\widetilde{U}(L_b)$ by including request of which the pickup or delivery node cannot be a predecessor of $v(L_b)$. Via the same reasoning as in Section 4.2.2 it can be derived that the latest possible time a vehicle could depart from a node to reach $v(L_b)$ on time is $t_d(L_b) = \max_{j \in P(L_b) \cup D(L_b)}\{\max\{t : t + \tau_{jv(L_b)}(t) \le t_l(L_b) - s_{v(L_b)}\}\}$. This means that any node j with the earliest possible departure time $e_j + s_j$ later than this time (i.e., $e_j + s_j > t_d(L_b)$) cannot be a predecessor of $v(L_b)$. Therefore, the corresponding request cannot be served anymore and can be added to the set $\widetilde{U}(L_b)$.

Furthermore, we define $\phi(L_b^1, L_b^2)$ (see Fig. 4) as:

$$\phi(L_b^1, L_b^2) = \max\{x \in R : \delta_{L_b^1}(t) + x \le \delta_{L_b^2}(t), \quad \forall t \in dom(L_b^2)\} \tag{56}$$

Then the dominance criterion will become:

Proposition 4.2. *Label L_b^2 is dominated by label L_b^1 if*

1. $v(L_b^1) = v(L_b^2)$
2. $U(L_b^1) \subseteq \widetilde{U}(L_b^2)$
3. $O(L_b^1) = O(L_b^2)$
4. $t(L_b^1) \ge t(L_b^2)$
5. $r(L_b^1) \ge r(L_b^2) - \phi(L_b^1, L_b^2)$
6. $q(L_b^1) \le q(L_b^2)$

For the proof of Proposition 4.2 the same reasoning as the proof of Proposition 4.1 could be followed.

4.3. Merging forward and backward labels

When all forward and backward labels are generated, they are merged to construct feasible profitable tours. A forward label L_f and a backward label L_b can be merged if the following conditions are satisfied:

1. $v(L_f) = v(L_b)$
2. $O(L_f) \cap O(L_b) = \{v(L_f)\}$
3. $(U(L_f) \setminus O(L_f)) \cap (U(L_b) \setminus O(L_b)) = \emptyset$
4. $q(L_f) + q(L_b) = q_{v(L_f)}$
5. $Img(L_f) \cap dom(L_b) \neq \emptyset$

The resulting path $p(L) = (p(L_f) \bigoplus p(L_b))$ has the following attributes:

1. $v(L) = 2n + 1$
2. $r(L) = r(L_f) + r(L_b) - r_{v(L_f)}$
3. $O(L) = \emptyset$
4. $U(L) = U(L_f) \cup U(L_b)$
5. $q(L) = 0$
6. $\delta_L(t) = \delta_{L_b}(\delta_{L_f}(t)), \forall t \in dom(L_f)$ such that $\delta_{L_f}(t) \in dom(L_b)$

However, this bidirectional labeling algorithm can generate duplicate solutions. Consider a feasible solution p^* including nodes i, j and k in this order. Each node $x \in p^*$ is associated with a forward label $L_f(x)$ and a backward label $L_b(x)$ (i.e., $v(L_f(x)) = v(L_b(x)) = x$). Therefore, the path p^* can be obtained by merging $L_f(i)$ with $L_b(i)$ as well as merging by $L_f(j)$ with $L_b(j)$. To overcome this drawback, we devised an additional test: we accept a solution only when a further extension of the forward label is impossible. In our example (see Fig. 5) the extension from $L_f(i)$ to node j is feasible and the extension from $L_f(j)$ to node k is infeasible by the predefined fixed time t_m. We generate solution p^* by merging $L_f(j)$ and $L_b(j)$ instead of $L_f(i)$ and $L_b(i)$. The test is performed for each candidate pair of labels and guarantees that each path is generated only once.

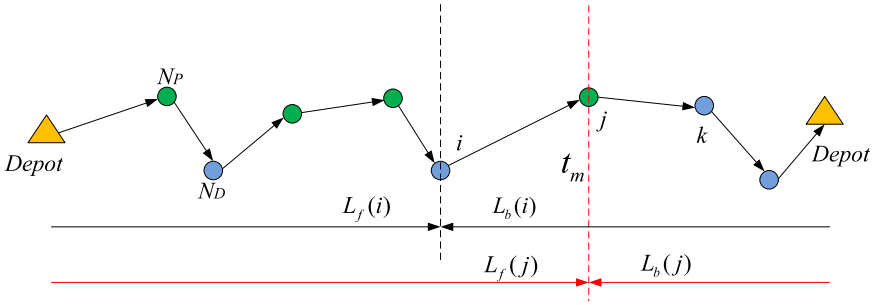

Fig. 5. An illustration of preventing duplicate solutions.

5. Restricted dynamic programing heuristic algorithm

Although the tailored labeling algorithm returns the optimal solution of our proposed problem, it is not fast enough to solve problems of realistic size within reasonable computation time.

In order to avoid enormous computation times and to save memory usage, Malandraki and Dial (1996) proposed a restricted dynamic programing heuristic algorithm for the TSP. The main concept is to limit in each stage the number of partial paths for further extension by a given parameter B. Furthermore, in each stage all partial paths should have the same number of visited nodes. Malandraki and Dial (1996) show that increasing the value of B results in better solutions, but also in substantial higher computation times. Note that $B = 1$ results in a nearest neighborhood heuristic and $B = \infty$ results it an exact dynamic programing algorithm.

Gromicho, van Hoorn, Kok, and Schutten (2012) propose a restricted dynamic programing algorithm for solving realistic VRPs and restrict the state space even further, by using a form of beam search (Bisiani, 1987), which means that each partial path is only expanded to E of its nearest feasible nodes.

The same principle of restricting the number of extension of a partial path can be applied to our labeling algorithm as well with some minor changes. Because of the precedence constraints, expanding a partial path to a pickup node may improve or deteriorate the objective function value depending on the profit of that node and the additional travel time to visit that node. However, expanding to a delivery node can only decrease the objective function value as additional travel time is necessary to visit that node while no profit is assigned the node. Moreover, because of the time-dependent travel time, the objective function value of a partial path is determined by its optimal departure time at the origin depot, which might change after a certain extension.

Therefore, it is not sufficient to select the partial paths with the highest objective function value, and we introduce a new selection method for each stage taking global information into account. For every extension of a partial path the $E/2$ best expanded partial paths ending with a pickup node and the $E/2$ best expanded partial paths ending with a delivery node are selected. Furthermore, in each stage we select the best $B/2$ partial paths that expand to a pickup node and the best $B/2$ partial paths that expand to a delivery node for further expansion. Moreover, instead of just using the objective function value for the selection, we use the earliest arrival time to order the expanded partial paths. If two expanded partial paths have the same earliest arrival time, the objective function value will be used to order them.

The restricted dynamic programing heuristic is described in Algorithm 1 . Herein, an empty labels set Z_k is created for each stage k for further extension (Step 2). Then the first label which represents the start from the depot is generated and put into the set Z_0 (Steps 3 and 4). If the number of the labels stored in each stage k is larger than B, only the B best labels are selected for further extension (Step 7). Moreover, for each label in stage k, only E of its feasible extensions are made and put in Z_{k+1} (Step 9). In the end, the best route Rt^* will be found by checking the labels in Z_{2n+1} (Step 10).

Note that there is no optimality guarantee anymore with such a restricted dynamic programing heuristic algorithm.

Algorithm 1: The restricted dynamic programing heuristic.

input : Request size n with two parameters B and E
output: The best route Rt^*

1 **for** *(each stage* $k = 0, 1, \ldots, 2n+1$*)* **do**
2 *Label set* $Z_k = \emptyset$
3 $L_0 \leftarrow \{v(L_0), \delta_{L_0}(t), O(L_0), U(L_0), q(L_0), r(L_0)\}$
4 $Z_0 \leftarrow Z_0 \cup \{L_0\}$
5 **for** *(each stage* $k = 0, 1, \ldots, 2n$*)* **do**
6 **if** *(Size of* $Z_k \geq B$*)* **then**
7 *Keep the B best labels in* Z_k
8 **for** *(each Label* $L_k \in Z_k$*)* **do**
9 *Do E of its feasible extensions. Put the non-dominated labels into* Z_{k+1}
10 *Find the best route* Rt^* *in set* Z_{2n+1}

Table 2
Characteristics of the TDPPDPTW and its variants instances.

Group	Q	W
AA	15	60
BB	20	60
CC	15	120
DD	20	120

6. Computational results

In this section the computational experiments are discussed. The algorithms are coded in JAVA and all computations are carried out on a single thread of a server with four CPU's (2.4 gigahertz/6 cores) and 64 gigabytes RAM. Each run has a time limit of 2 weeks and a maximum memory allowance of 16 gigabytes RAM. For our numerical study we use an adapted version of the instances for the PDPTW in Ropke et al. (2009). In these instances, the coordinates of each pickup and delivery location are both randomly and uniformly distributed over a $[0, 50] \times [0, 50]$ square and a single depot is located in this square. The load q_i for each request R_i is also randomly selected from the interval $[5, Q]$, where Q is the vehicle capacity. Table 2 summarizes the characteristic of these instances including the vehicle capacity Q and the width of the time windows W. For each group, there are 10 instances with the number

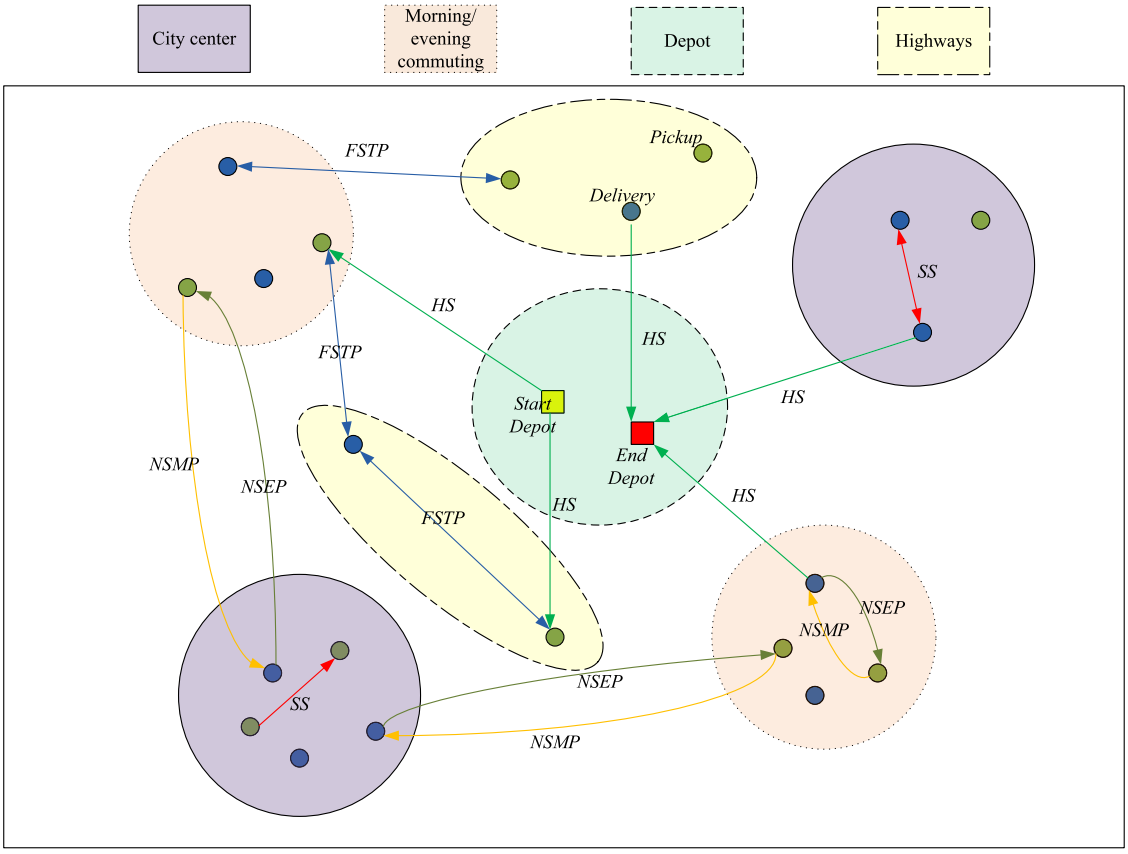

Fig. 6. An illustration of a instance with different speed profiles.

of requests ranging from 30 to 75. For more details, we refer to Ropke et al. (2009).

In addition to the instances proposed by Ropke et al. (2009), we created for each class four new small instances with the amount of requests ranging from 10 to 25. They are generated by only keeping the first α requests of the instance with 30 requests of each class, where $\alpha = 10$, 15, 20 and 25. For example, the instance AA10 is created by considering the first ten requests of AA30.

In our instances, the coordinates, time windows and loads of the depot, pickup nodes and delivery nodes are the same as in the instances of Ropke et al. (2009). Furthermore, the vehicle capacity Q is also the same. In the instances of Ropke et al. (2009) the travel time is fixed and no profits were assigned to the requests.

To make the travel time time-dependent, road congestion is handled by a so-called speed model which consists of different speed profiles. It is used to determine the travel time between two nodes on a specific departure time. This speed model is based on the speed model of Verbeeck et al. (2014) for the TDOP and Dabia et al. (2013) for the TDVRPTW. Without loss of generality, we assume that breakpoints are the same for all speed profiles as congestion tends to happen around the same time regardless of the type of speed profile. The pickup and delivery node of each request is randomly assigned to one of the three predefined areas: morning and evening commuting area, city center and highways. Then, the speed profile of a link is assigned according to the type (e.g., depot or request node) and location of the tail node and head node.

In our speed model, the planning horizon covers 14 working hours (840 minutes, from 7 a.m. to 9 p.m.) and a minute is set to be one unit of time, while, in Ropke et al. (2009), the planning horizon of length 600 minutes is considered. Each speed profile has four non-overlapping time periods with constant speed, reflecting two congested periods and two periods with normal traffic conditions. Five speed profiles are included (see Fig. 6 and Table 3):

Table 3
Speed profiles.

Congestion description Time periods	Morning peak 7 a.m.–9 a.m.	Normal 9 a.m.–5 p.m.	Evening peak 5 p.m.–7 p.m.	Normal 7 p.m.–9 p.m.
1. SS	0.5	0.81	0.5	0.81
2. NSMP	0.67	1.33	0.88	1.33
3. NSEP	0.88	1.33	0.67	1.33
4. FSTP	0.85	1.5	0.85	1.5
5. HS	1.0	2.0	1.0	2.0

- Slow speed (SS): these links represent a busy central business district (CBD) with a lot of traffic during the whole day.
- Normal speed with morning peak (NSMP): these links represent roads leading from a residential area to the CBD. These roads are in most cases congested in the morning.
- Normal speed with evening peak (NSEP): these links represent roads leading from a CBD to a residential zone. The roads typically encounter evening congestion.
- Fast speed with two peaks (FSTP): these links represent roads near the highway with a morning and evening peak in both directions.
- High speed (HS): these links connect the request nodes with the depot .

As in Ropke et al. (2009) all requests needed to be served no profits were assigned to the requests. To come up with meaningful settings for the profits we did preliminary tests with 20, 40 and 80 units of profit assigned to each pickup node. As shown in Table 4, if the profit of the requests is 20, it will be too low. There are even

Table 4
Comparison of results for different profits settings.

Instance	20 units profit for each request			40 units profit for each request			80 units profit for each request		
	Optimal value	Profits (served)	Traveling cost	Optimal value	Profits (served)	Traveling cost	Optimal value	Profits (served)	Traveling cost
AA10	0	0 (0)	0	27.14	80 (2)	52.86	196.22	400 (5)	203.78
AA15	0	0 (0)	0	37.77	120 (3)	82.23	392.37	880 (11)	487.63
AA20	0	0 (0)	0	73.7	240 (6)	166.3	566.03	1200 (15)	633.97
AA25	9.06	60 (3)	50.94	203.01	560 (14)	356.99	884.41	1440 (18)	555.59
AA30	13.07	60 (3)	46.93	266.72	640 (16)	373.28	1020.86	1600 (20)	579.14
BB10	0	0 (0)	0	39.26	120 (3)	80.74	247.65	720 (9)	472.35
BB15	6.61	20 (1)	13.39	99.58	280 (7)	180.42	518.03	960 (12)	441.97
BB20	6.61	20 (1)	13.39	138.05	600 (15)	461.95	738.05	1200 (15)	461.95
BB25	6.25	40 (2)	33.75	147.88	520 (13)	372.12	787.64	1360 (17)	572.36
BB30	4.26	20 (1)	15.74	274.45	840 (21)	565.55	1114.45	1680 (21)	565.55
CC10	2.81	40 (2)	37.19	57.63	160 (4)	102.37	340.2	560 (7)	219.8
CC15	5.69	40 (2)	34.31	98.29	280 (7)	181.71	469.59	960 (12)	490.41
CC20	4.35	20 (1)	15.65	97.27	480 (12)	382.73	712.53	1280 (16)	567.47
CC25	9.42	60 (3)	50.58	226.23	720 (18)	493.77	984.3	1600 (20)	615.7
CC30	3.73	60 (3)	56.27	316.39	800 (20)	483.61	1167.47	1760 (22)	592.53
DD10	11.8	60 (3)	48.2	85.88	160 (4)	74.12	315.34	720 (9)	404.66
DD15	10.18	60 (3)	49.82	99.05	200 (5)	100.95	554.55	1040 (13)	485.45
DD20	14.44	80 (4)	65.56	182.14	640 (16)	457.86	832.05	1360 (17)	527.95
DD25	6.66	80 (4)	73.34	250.16	680 (17)	429.84	1025.75	1600 (20)	574.25
DD30	16.9	80 (4)	63.1	343.65	840 (21)	496.35	1247.22	1840 (23)	592.78

Table 5
Comparison of the tailored labeling algorithm without and with dominance and feasibility check.

Instance	TDTL without dominance and feasibility check				TDTL with dominance and feasibility check					
	Optimal value	Generated labels (#)	Generated routes (#)	Time (seconds)	Optimal value	Generated labels (#)	Generated routes (#)	Dominated labels (#)	Infeasible labels (#)	Time (seconds)
AA10	27.14	1680	19	0.47	27.14	196	15	121	0	0.23
AA15	37.77	67,201	57	1.00	37.77	471	26	329	0	0.37
AA20	73.7	1,295,435	658	12.04	73.7	1183	49	886	12	0.80
AA25	–	–	–	Out of memory	203.01	10,174	80	8113	974	1.59
AA30	–	–	–	Out of memory	266.72	18,719	77	15,012	2247	2.79
BB10	39.26	3647	43	0.37	39.26	204	21	104	0	0.22
BB15	99.58	387,803	5311	3.93	99.58	1729	52	1023	263	0.50
BB20	138.05	14,067,769	420,587	30019.42	138.05	11,572	89	9445	1565	1.95
BB25	–	–	–	Out of memory	147.88	13,787	145	10,660	1886	2.25
BB30	–	–	–	Out of memory	274.45	40,335	152	31,755	7019	3.17
CC10	57.63	4859	70	0.44	57.63	289	25	184	0	0.26
CC15	98.29	119,687	1455	1.92	98.29	1799	43	1501	19	0.62
CC20	97.27	5,763,951	14,981	47.21	97.27	5206	82	4337	215	1.25
CC25	–	–	–	86,400	226.23	97,782	278	81,472	6626	11.95
CC30	–	–	–	Out of memory	316.39	347,803	490	284,760	42,498	78.94
DD10	85.88	5567	55	0.61	85.88	222	20	129	0	0.25
DD15	99.05	1,041,425	2272	5.05	99.05	1471	53	850	267	0.61
DD20	–	–	–	Out of memory	182.14	11,292	111	8473	1735	2.11
DD25	–	–	–	Out of memory	250.16	405,617	630	324,304	48,709	199.73
DD30	–	–	–	Out of memory	343.65	1,257,385	547	1,085,122	128,811	855.20

routes in which it is best to do nothing. If the profit per request is 40, on average 56% of the requests are served and the total profit in the route is 52% higher than the traveling cost. When the profits increase to 80, the provider would like to serve more request (on average 74%) and the generated profits are way (on average 141%) higher than the traveling cost. To avoid bias on either profits or traveling cost, in the remainder of the computational experiments, we decided to assign a profit of 40 units to all pickup nodes, which makes a request only profitable if the additional travel time to serve the request (i.e., visiting both the pickup and delivery node) is less than 40 time units.

6.1. Performance of the tailored labeling algorithm

Table 5 gives the results of the experiments using the tailored labeling algorithm without dominance and feasibility check and the tailored labeling algorithm with dominance and feasibility check. In Table 5, we observe that the tailored labeling algorithm without dominance and feasibility check can only solve 11 instances (out of 20 instances) to optimality within the given memory limit (16 gigabytes) or time limit (1 day). In contrast, the tailored labeling algorithm with dominance and feasibility check is able to solve all of the instances within a reasonable running time (within 15 minutes). Moreover, due to the dominance and feasibility check much less labels and routes (columns 7 and 8) are generated compared to its counterpart (columns 3 and 4). Therefore, it shows that the proposed dominance criterion and feasibility check both show their great potentials to get rid of the unpromising labels (columns 9 and 10) which are proven not to be the part of the optimal solution. On average 84% of the labels are dominated and about 11% is proven to lead to an infeasible extension.

In Table 6, we present the results of the experiments using the mathematical model introduced in Section 3.2, solved by the optimization software Gurobi (version 5.6) with its default parame-

Table 6
Results of solving the mathematical model by using the Gurobi.

Instances	LB	Best	Time (seconds)	Gap (%)	Gap_opt (%)
AA10	27.14	27.14	1.95	0	0
AA15	37.77	37.77	4.66	0	0
AA20	73.7	73.7	24.93	0	0
AA25	203.01	203.01	46344.06	0	0
AA30	216.34	543.47	86,400	151	50.92
BB10	39.26	39.26	1.451	0	0
BB15	99.58	99.58	121.03	0	0
BB20	138.05	138.05	1669.74	0	0
BB25	147.88	147.88	48616.51	0	0
BB30	245.7	515.404	86,400	110	46.75
CC10	57.63	57.63	3.29	0	0
CC15	98.29	98.29	44.34	0	0
CC20	97.27	97.27	7935.29	0	0
CC25	209.08	1316.69	86,400	530	82.82
CC30	237.02	1674.74	86,400	607	81.11
DD10	85.88	85.88	1.43	0	0
DD15	99.05	99.05	54.65	0	0
DD20	182.14	182.14	4959.48	0	0
DD25	221.39	1419.81	86,400	541	82.38
DD30	279.51	1733.62	86,400	520	80.18

ter settings and a computation time limit of 1 day. For each instance, we provide the lower bound (LB), best feasible solution found (Best), the processing time given in seconds (Time) and the gap (*Gap*) provided by Gurobi and the relative gap with respect to the optimal solution (Gap_{opt}). The relative gap is calculate by $Gap_{opt} = 100 \times \frac{Best-Opt}{Best}$ in which *Opt* is the value of the optimal solution for the instance derived by our tailored labeling algorithm. As shown in Table 6, the proposed mathematical model is able to solve instances with up to 25 requests within the time limit. However, for larger instances, it takes already more than one day compared to the at maximum 15 minutes of our tailored labeling algorithm.

6.2. Results of the tailored labeling algorithm on all instances

In Table 7, we report the results of our tailored labelling algorithm. We present the computation times for both the bidirectional search (i.e., $t_m = 420$, the middle of the time horizon) and the mono-directional search (i.e., $t_m = 840$, the end of the time horizon). Since both algorithms lead to the optimal value, this value is presented only once (column 2). The bidirectional search shows greater potential power in terms of computation time. For some instances (DD40 and DD50), the bidirectional search is even 10 times faster than the monodirectional search. Moreover, the bidirectional search is capable of finding the optimal value of more instances than the monodirectional search within our set computation time limit of 2 weeks. This is mainly because of the fact that the number of labels needed to be processed in the bidirectional search is considerably less compared to the mono-directional search. However, merging the forward and backward labels also requires some computational effort. It can be seen that for 6 easy instances (AA35, BB30, BB40, BB50, BB55, BB70) the extra time necessary to merge the labels outweighs the reduced time to generate labels in the bidirectional search. As we prefer an approach which is also able to solve difficult instances we focus in the remainder of this paper at the bidirectional search.

With the bidirectional search, 34 of the 40 instances could be solved within our time limit of 2 weeks. We did not find optimal solutions for the instances with more than 65 requests in the CC group and more than 55 requests in the DD group. This means that the algorithm was still generating labels after 2 weeks, and therefore also no upper bound is available.

It is not a surprise that the computation time increases with the number of requests. However, Table 7 also demonstrates that the computation time increases if the vehicle capacity (i.e., BB compared with AA and DD compared with CC) or time window width (CC compared with AA and DD compared with BB) increases. This is mainly caused as a larger vehicle capacity or time window width will lead to a larger solution space.

Table 7
Monodirectional algorithm vs. bidirectional algorithm.

Instance	Optimal value	Requests served	Processing time (seconds)	
			Monodirectional	Bidirectional
AA30	266.72	16	2.37	2.12
AA35	278.14	20	4.68	5.04
AA40	341.74	22	7.94	5.82
AA45	361.57	21	23.54	5.6
AA50	430.41	23	22.40	9.53
AA55	436.05	27	32.79	18.66
AA60	505.02	26	87.11	24.71
AA65	576.39	29	182.09	34.80
AA70	606.68	29	180.01	39.14
AA75	715.53	31	1038.67	257.60
BB30	274.45	21	2.36	2.78
BB35	337.55	22	1.30	1.03
BB40	326.82	23	8.30	9.95
BB45	378.90	23	34.88	27.94
BB50	432.10	25	31.81	37.41
BB55	530.84	27	40.14	59.02
BB60	558.02	28	164.38	119.2
BB65	547.84	25	353.15	332.13
BB70	558.29	27	225.79	351.57
BB75	613.35	27	775.58	511.37
CC30	316.39	20	32.51	27.64
CC35	386.26	24	133.15	104.18
CC40	464.28	25	2934.28	942.27
CC45	497.68	26	20641.34	3422.21
CC50	519.60	26	45532.40	13913.43
CC55	581.50	28	156770.32	28429.25
CC60	624.95	30	$\geq$ 2 weeks	52801.65
CC65	663.31	31	$\geq$ 2 weeks	699831.05
DD30	343.65	21	396.23	80.42
DD35	410.19	22	2815.31	338.16
DD40	490.92	25	53402.91	4618.11
DD45	540.17	27	67293.53	7667.97
DD50	610.07	30	186697.41	21889.86
DD55	639.75	29	$\geq$ 2 weeks	823755.21

In Table 8 we present a more detailed look at the solution of instance AA30. The solution only serves 16 requests in the reported sequence. Since the time dependent travel time instead of the traveled distance is considered as part of the objective, the vehicle's departure time becomes crucial. Therefore, delaying the departure time of the vehicle may lead to less traveling cost. We observe that the vehicle departs from the origin depot at 208.23 and arrives to the destination depot at 581.51. Other departure times will lead to higher travel cost.

In Table 9, we also present the results on small instances if we double the capacity of the vehicles. On one hand, vehicles with more capacity can serve more requests, which returns a better objective than its less capacitate counterparts. On the other hand, for some groups, the instances with larger vehicle capacities need much more time to get the optimal solution (e.g., CC25 and DD25).

6.3. Performance of the restricted dynamic programing heuristic

We also have conducted computational experiments to analyze the solution quality produced by the restricted dynamic programing heuristic. In Table 7, we have seen that the exact procedure has enormous computation times. In Table 10, we show the results for the same instances but now with the restricted dynamic programing heuristic. The restricted dynamic programing heuristic algorithm has been run with different values for *B*. In these first tests

Table 8
Solution of AA30.

Start time	End time	Travel cost	Served request	Generated profit
208.23	581.51	373.28	1, 11, 3, 26, 22, 18, 28, 25, 19, 8, 5, 13, 15, 7, 4, 27	640

Table 9
Comparison of the different capacity settings.

Instances	Original Ropke's capacity					Double Ropke's capacity				
	Optimal value	Profits	Requests served	Travel cost	Time (seconds)	Optimal value	Profits	Requests served	Traveling cost	Time (seconds)
AA10	27.14	80	2	52.86	0.23	27.14	80	2	52.86	0.30
AA15	37.77	120	3	82.23	0.37	79.63	200	5	120.37	0.95
AA20	73.7	240	6	166.3	0.80	146.41	440	11	293.59	2.57
AA25	203.01	560	14	356.99	1.59	286.68	600	15	313.32	29.47
BB10	39.26	120	3	80.74	0.22	46.49	200	5	153.51	0.48
BB15	99.58	280	7	180.42	0.50	131.25	320	8	188.75	2.65
BB20	138.05	600	15	461.95	1.95	192.46	600	15	407.54	11.78
BB25	147.88	520	13	372.12	2.25	268.91	800	20	531.09	20.72
CC10	57.63	160	4	102.37	0.26	96.56	240	6	143.44	0.62
CC15	98.29	280	7	181.71	0.62	121.9	320	8	198.1	2.74
CC20	97.27	480	12	382.73	1.25	147.50	520	13	372.49	5.33
CC25	226.23	720	18	493.77	11.95	434.01	960	24	525.99	10808.19
DD10	85.88	160	4	74.12	0.25	99.13	160	4	60.87	0.31
DD15	99.05	200	5	100.95	0.61	108.55	200	5	91.45	1.37
DD20	182.14	640	16	457.86	2.11	233.61	640	16	406.39	19.30
DD25	250.16	680	17	429.84	199.73	452.94	960	24	507.06	82620.40

Table 10
Impact of different B values on instances in Table 7.

		B=1000			B=2500			B=5000			B=10,000		
Instance	Optimal	Value	Time (seconds)	Gap	Value	Time (seconds)	Gap	Value	Time (seconds)	Gap	Value	Time (seconds)	Gap
AA30	266.72	266.72	3.32	0.00	266.72	3.03	0.00	266.72	3.14	0.00	266.72	3.01	0.00
AA35	278.14	278.14	4.43	0.00	278.14	4.17	0.00	278.14	4.10	0.00	278.14	4.31	0.00
AA40	341.74	341.74	4.06	0.00	341.74	5.16	0.00	341.74	5.09	0.00	341.74	4.23	0.00
AA45	361.58	361.58	6.65	0.00	361.58	7.40	0.00	361.58	7.68	0.00	361.58	7.89	0.00
AA50	430.41	430.41	8.22	0.00	430.41	11.93	0.00	430.41	12.92	0.00	430.41	14.29	0.00
AA55	436.05	433.21	11.33	0.65	436.05	15.29	0.00	436.05	18.42	0.00	436.05	23.25	0.00
AA60	505.02	505.02	11.19	0.00	505.02	16.68	0.00	505.02	21.76	0.00	505.02	26.66	0.00
AA65	576.39	576.39	14.87	0.00	576.39	21.78	0.00	576.39	27.04	0.00	576.39	27.25	0.00
AA70	606.68	606.68	15.05	0.00	606.68	25.12	0.00	606.68	30.13	0.00	606.68	33.98	0.00
AA75	715.53	714.65	32.07	0.12	714.65	57.36	0.12	715.53	111.54	0.00	715.53	162.37	0.00
BB30	274.45	274.45	2.00	0.00	274.45	1.89	0.00	274.45	1.88	0.00	274.45	1.86	0.00
BB35	337.55	337.55	1.03	0.00	337.55	1.21	0.00	337.55	1.05	0.00	337.55	1.17	0.00
BB40	326.82	326.82	4.01	0.00	326.82	4.79	0.00	326.82	5.07	0.00	326.82	5.37	0.00
BB45	378.90	378.90	8.61	0.00	378.90	12.89	0.00	378.90	17.43	0.00	378.90	19.14	0.00
BB50	432.10	432.10	8.04	0.00	432.10	14.40	0.00	432.10	19.88	0.00	432.10	22.75	0.00
BB55	530.84	530.84	12.40	0.00	530.84	20.19	0.00	530.84	26.80	0.00	530.84	33.32	0.00
BB60	558.02	558.02	16.08	0.00	558.02	28.47	0.00	558.02	43.92	0.00	558.02	69.61	0.00
BB65	547.84	547.84	16.41	0.00	547.84	31.09	0.00	547.84	61.08	0.00	547.84	104.54	0.00
BB70	558.29	558.29	21.72	0.00	558.29	34.10	0.00	558.29	50.89	0.00	558.29	83.74	0.00
BB75	613.35	613.35	23.32	0.00	613.35	45.66	0.00	613.35	65.52	0.00	613.35	106.88	0.00
CC30	316.39	316.39	9.16	0.00	316.39	13.39	0.00	316.39	16.18	0.00	316.39	18.46	0.00
CC35	386.26	386.26	17.39	0.00	386.26	31.84	0.00	386.26	38.27	0.00	386.26	53.09	0.00
CC40	464.28	464.28	32.67	0.00	464.28	56.99	0.00	464.28	85.19	0.00	464.28	124.12	0.00
CC45	497.68	497.68	52.17	0.00	497.68	108.30	0.00	497.68	190.84	0.00	497.68	303.98	0.00
CC50	519.60	516.59	48.71	0.58	516.59	171.45	0.58	517.10	364.55	0.48	517.10	616.42	0.48
CC55	581.50	564.03	98.13	3.00	572.88	306.60	1.48	572.88	746.83	1.48	581.50	1416.82	0.00
CC60	624.95	593.46	114.55	5.04	606.11	327.94	3.01	624.95	911.43	0.00	624.95	1840.42	0.00
CC65	663.31	636.12	169.39	4.10	653.51	470.64	1.48	655.84	1295.07	1.13	662.61	3879.84	0.11
CC70	–	678.11	188.24	$\geq$ 0.55	680.30	585.96	$\geq$ 0.23	680.56	1589.12	$\geq$ 0.19	681.84	4858.95	$\geq$0.00
CC75	–	686.68	250.89	$\geq$ 0.00	686.68	634.38	$\geq$ 0.00	686.68	1745.03	$\geq$ 0.00	686.68	7159.92	$\geq$0.00
DD30	343.65	343.65	17.32	0.00	343.65	35.09	0.00	343.65	46.71	0.00	343.65	46.1	0.00
DD35	410.19	410.19	37.57	0.00	405.07	92.01	1.25	410.19	168.55	0.00	410.19	275.19	0.00
DD40	490.92	477.82	74.59	2.67	490.92	233.52	0.00	490.92	526.53	0.00	490.92	1026.12	0.00
DD45	540.17	540.17	78.77	0.00	540.17	210.87	0.00	540.17	415.69	0.00	540.17	953.24	0.00
DD50	610.07	589.67	104.49	3.34	599.02	264.49	1.81	610.07	725.41	0.00	610.07	1277.29	0.00
DD55	639.75	617.52	159.5	3.47	626.86	550.42	2.01	629.70	1132.39	1.57	639.75	2754.41	0.00
DD60	–	690.87	191.22	$\geq$ 3.16	695.74	742.63	$\geq$ 2.47	695.54	1785.51	$\geq$ 2.50	713.39	4054.79	$\geq$ 0.00
DD65	–	754.65	246.71	$\geq$ 0.75	758.57	963.77	$\geq$ 0.23	760.32	2212.53	$\geq$ 0.00	760.32	5853.15	$\geq$ 0.00
DD70	–	746.44	334.40	$\geq$ 2.31	751.56	1434.89	$\geq$ 1.64	753.58	3869.98	$\geq$ 1.38	764.10	9581.94	$\geq$ 0.00
DD75	–	830.50	399.73	$\geq$ 0.63	831.49	1899.06	$\geq$ 0.51	835.76	4816.39	$\geq$ 0.00	835.76	15842.28	$\geq$ 0.00
Average	515.18	510.34	71.26	$\geq$ 0.76	512.48	237.42	$\geq$ 0.42	513.72	580.44	$\geq$ 0.22	515.10	1567.30	$\geq$0.01

Table 11
Impact of different E values on instances when B=10,000.

	E=0.5n			E=0.25n			E=0.125n		
Instance	Value	Time (seconds)	Gap	Value	Time (seconds)	Gap	Value	Time (seconds)	Gap
AA30	266.72	4.23	0.00	258.68	3.99	3.01	183.89	1.33	31.06
AA35	278.14	5.01	0.00	278.14	4.73	0.00	268.40	2.14	3.50
AA40	341.74	7.27	0.00	341.74	6.93	0.00	341.74	4.57	0.00
AA45	361.58	8.88	0.00	361.58	8.49	0.00	358.74	8.22	0.79
AA50	430.41	20.11	0.00	430.41	11.01	0.00	418.09	9.19	2.86
AA55	436.05	33.49	0.00	436.05	26.04	0.00	424.6	13.15	2.63
AA60	505.02	31.72	0.00	505.02	23.89	0.00	505.02	14.34	0.00
AA65	576.39	30.23	0.00	576.39	24.45	0.00	568.4	22.18	1.39
AA70	606.68	79.56	0.00	606.68	47.77	0.00	599.97	31.50	1.11
AA75	715.53	294.76	0.00	715.53	210.50	0.00	713.67	144.16	0.26
BB30	274.45	3.53	0.00	274.45	2.20	0.00	249.00	1.48	9.27
BB35	337.55	2.62	0.00	316.85	5.02	6.13	316.85	1.89	6.13
BB40	326.82	6.6	0.00	326.82	6.52	0.00	326.82	4.54	0.00
BB45	378.90	25	0.00	378.9	14.61	0.00	378.9	11.62	0.00
BB50	432.10	32.89	0.00	432.10	17.19	0.00	432.1	11.83	0.00
BB55	530.84	50.25	0.00	530.84	31.97	0.00	530.84	18.36	0.00
BB60	558.02	72.85	0.00	558.02	71.65	0.00	558.02	36.90	0.00
BB65	547.84	129.72	0.00	547.84	120.02	0.00	547.84	56.83	0.00
BB70	558.29	139.86	0.00	558.29	92.68	0.00	558.29	48.46	0.00
BB75	613.35	190.45	0.00	613.35	158.15	0.00	613.35	49.19	0.00
CC30	316.39	20.66	0.00	316.39	19.25	0.00	292.65	5.80	7.50
CC35	386.26	54.93	0.00	386.26	45.49	0.00	375.17	12.00	2.87
CC40	464.28	133.99	0.00	464.28	152.86	0.00	448	38.50	3.51
CC45	497.68	497.68	0.00	497.68	352.53	0.00	479.19	191.45	3.72
CC50	517.1	542.93	0.48	517.1	598.14	0.48	503.47	421.23	3.10
CC55	581.5	1920.55	0.00	581.5	1421.72	0.00	579.34	2338.11	0.37
CC60	624.95	1244.56	0.00	624.95	1579.68	0.00	624.95	2338.36	0.00
CC65	662.61	4098.04	0.11	662.61	2629.06	0.11	662.61	2203.57	0.11
CC70	681.84	4918.55	0.00	681.84	3110.90	0.00	681.84	2453.23	0.00
CC75	686.68	4290.45	0.00	686.68	5518.81	0.00	686.68	3183.47	0.00
DD30	343.65	68.19	0.00	343.65	63.74	0.00	330.79	17.25	3.74
DD35	410.19	350.53	0.00	410.19	399.22	0.00	385.85	84.60	5.93
DD40	490.92	1158.38	0.00	490.92	1133.28	0.00	483.48	1036.05	1.52
DD45	540.17	889.37	0.00	540.17	962.77	0.00	532.17	827.67	1.48
DD50	610.07	1919.31	0.00	610.07	1595.29	0.00	597.37	1352.25	2.08
DD55	639.75	2861.15	0.00	639.75	2733.71	0.00	638.36	4879.35	0.22
DD60	713.39	5432.74	0.00	713.39	4656.094	0.00	708.52	4264.71	0.68
DD65	760.32	6456.7	0.00	760.32	5255.72	0.00	758.57	4545.88	0.23
DD70	764.1	9593.25	0.00	764.1	8355.49	0.00	760.5	6054.51	0.47
DD75	835.76	11762.83	0.00	835.76	10770.09	0.00	835.76	8465.89	0.00
Average	515.10	1484.60	0.01	514.38	1306.04	0.15	506.50	1130.14	1.69

we set $E = \infty$. For the instances of which the optimal solution is not known, we compare the solution to the best found solution (i.e., the solution found in case $B = 10,000$) to have a lower bound on the gap.

With a value of $B = 1000$ the algorithm leads in on average 71.26 seconds to the optimal solution for 25 out of the 40 instances. For 1 instance it is not sure whether or not the optimal solution is reached. The average gap is at least 0.79%. With a value of $B = 2500$ the average computation time increased to 237.42 seconds. For one more instance the optimal solution is found, and again there is 1 instance for which it is not sure whether or not the optimal solution is reached. The average gap (at least 0.42%) is about half of the gap as in the case of $B = 1000$. With a value of $B = 5000$ the average gap is halved again to at least 0.22% and the number of instances for which the optimal solution is found increases considerably to 30. For 3 more instances the solution found could be the optimal one. However, the average computation time is also increased to 580.44 seconds. Increasing the value of B to 10,000 ensures that 32 of the 34 instances of which we know the optimal solution are solved to optimality. The average computation time in case $B = 10,000$ equals 1567.3 seconds. The two instances which were not solved to optimality have, respectively, a gap of 0.48% and 0.11%. This means that, compared to the tailored labeling algorithm, the average gap of the restricted dynamic programing heuristic algorithm over the 34 instances of which the optimal solution is known is just 0.02%.

Note that all the instances for which the exact labeling algorithm was not able to solve in 2 weeks (e.g., *DD*75 in Table 7), can be solved within 20 hours with this restricted dynamic programing algorithm.

In Table 11, we present the impact of different E values on the performances of the algorithm with $B = 10,000$. We set the values of E to $0.5n$, $0.25n$ and $0.125n$ and fractional values are rounded to its closed integer.

The results indicate that the computation times decrease if the value for E decreases, but not that much. As can be seen in Table 11, when the value of E is equal to $0.5n$, the solution quality is the same as in case $E = \infty$ but the computation times are reduced by 5%. When we restrict E further the solution quality is going down drastically without gaining too much computation time. In comparison, a combination of $B = 1000$ and $E = \infty$ is much faster and has higher solution quality then the combination of $B = 10,000$ and $E = 0.125n$

7. Conclusion

This paper presents the time-dependent capacitated profitable tour problem with time windows and precedence constraints, which combines features of both the traveling salesman problem

with pickup and delivery, and the traveling salesman problem with profits. Moreover, time-dependent traveling speeds are considered to capture road congestion, which increases the complexity of the problem.

We propose a tailored labeling algorithm to solve this problem enriched by new and strong dominance rules to discard 95% of the labels. Extensive computational results show that most instances with up to 75 requests can be solved to optimality within the given time limit of two weeks, but also show that some instances remain unsolved.

To reduce the computation time and memory usage, a restricted dynamic programing heuristic algorithm is implemented. The heuristic is able to find solutions for all instances with good qualities (on average 0.01% gap) and less computational time (on average 1567 seconds).

Obviously, the performance of our exact algorithm critically depends on the tailored dominance criterion that we propose. It shows great potential when there are relatively tight time windows attached to all the nodes. Further strengthening dominance is a promising direction for future research.

In addition, solving extensions of the time-dependent capacitated profitable tour problem with time windows and precedence constraints seems to be an attractive research direction. More specifically, the time-dependent variant of the pickup and delivery problem with time windows (TDPDPTW) and the time-dependent team orienteering problem (TDTOP) are very interesting as it aims to optimize the routes of a fleet of vehicles, instead of a single vehicle only. When column generation is utilized to solve those two problems in an exact way, our proposed model can be used as the pricing problem.

Acknowledgments

The authors gratefully acknowledge funds provided by China Scholarship Council (CSC) under Grant No. 201206160092.

References

Balas, E. (1989). The prize-collecting traveling salesman problem. *Networks, 19*(6), 621–636.
Bisiani, R. (1987). Beam search. In S. Shapiro (Ed.), *Encyclopedia of artificial intelligence* (pp. 56–58). Wiley and Sons.
Dabia, S., Ropke, S., van Woensel, T., & de Kok, T. (2013). Branch and cut and price for the time dependent vehicle routing problem with time windows. *Transportation Science, 47*(3), 380–396.
Dell' Amico, M., Maffioli, F., & Varbrand, P. (1995). On prize-collecting tours and the asymmetric traveling salesman problem. *International Transactions in Operational Research, 2*(3), 297–308.
Donati, A., Montemanni, R., Casagrande, N., Rizzolo, A. E., & Gambardella, L. (2008). Time dependent vehicle routing problem with a muti-ant colony system. *European Journal of Operational Research, 185*(3), 1174–1191.
Dumitrescu, I., Ropke, S., & Cordeau, J. (2010). The traveling salesman problem with pickup and delivery: polyhedral results and branch-and-cut algorithm. *Mathematic Programming Series A, 121*(2), 269–305.
Feillet, D., Dejax, P., & Gendreau, M. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks, 44*(3), 216–229.
Feillet, D., Dejax, P., & Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science, 39*(2), 188–205.
Fomin, F., & Lingas, A. (2002). Approximation algorithms for time-dependent orienteering. *Information Processing Letters, 83*(2), 57–62.
Gromicho, J., van Hoorn, J., Kok, A., & Schutten, J. (2012). Restricted dynamic programming: A flexible framework for solving realistic VRPs. *Computers and Operations Research, 39*(5), 902–909.
Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research, 255*(2), 315–332.
Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., & Yagiura, M. (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematics, 156*(11), 2050–2069.
Ichoua, S., Gendreau, M., & Potvin, J. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research, 144*(2), 379–396.
Laporte, G., & Martello, S. (1990). The selective traveling salesman problem. *Discrete Applied Mathematics, 26*(2–3), 193–207.
Li, J. (2011). Model and algorithm for time-dependent team orienteering problem. In S. Lin, & X. Huang (Eds.), *Advanced research on computer education, simulation and modeling. Communications in computer and information science: vol. 175* (pp. 1–7).
Malandraki, C., & Daskin, M. (1992). Time dependent vehicle routing problems: Formulations, properties, and heuristic algorithms. *Transportation Science, 26*(3), 185–200.
Malandraki, C., & Dial, R. (1996). A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research, 90*(1), 45–55.
Righini, G., & Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization, 3*(3), 255–273.
Ropke, S., Cordeau, J.-F., & Laporte, G. (2009). Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science, 43*(3), 267–286.
Ruland, K., & Rodin, E. (1997). The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers Mathematics with Applications, 33*(12), 1–13.
Ruland, K. S. (1994). *Polyhedral solution to the pickup and delivery problem*. Ph.D. thesis, Sever Institute, Washington University in St.Louis.
Sigurd, M., & Pisinger, D. (2004). Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science, 38*(2), 197–209.
Sol, M. (1994). *Column generation techniques for pickup and delivery problems*. Ph.D thesis, Technische Universiteit Eindhoven.
Taş, D., Dellaert, N., van Woensel, T., & de Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times. *Transportation Research Part C: Emerging Technologies, 48*, 66–83.
Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: a survey. *European Journal of Operational Reseach, 209*(1), 1–10.
Verbeeck, C., Aghezzaf, E. H., & Vansteenwegen, P. (2014). A fast solution method for the time-dependent orienteering problem with time windows. *European Journal of Operational Research, 236*(2), 419–432.
Verbeeck, C., Vansteenwegen, P., & Aghezzaf, E.-H. (2016). Solving the stochastic time-dependent orienteering problem with time windows. *European Journal of Operational Research, 255*(3), 699–718.
van Woensel, T., Kerbache, L., Peremans, H., & Vandaele, N. (2008). Vehicle routing with dynamic travel times: A queueing approach. *European Journal of Operational Research, 186*(3), 990–1007.